Curing the Software Requirements And Cost Estimating Blues

The Fix is Easier Than You Might Think

MAJ. MIKE NELSON, U.S. ARMY
JAMES CLARK • MARTHA ANN SPURLOCK

Software plays an ever-increasing role in the operation of Department of Defense (DoD) weapon systems; Command, Control, Communications, Computers & Intelligence (C4I) systems; and management information systems. As Figure 1 depicts, the size of software in today's DoD weapon systems is quite large, and due to the increasing digitization of the battlefield, the numbers will only increase. For example, since 1980 the Army's inventory of source lines of code has increased from approximately 5 million to over 100 million.[1] Not only is the size of our software increasing, but as Figure 2 illustrates, our dependence on software is also increasing.

Extrapolating from a September 1994 report released by the Electronic Industries Association,[2] by the year 2000 DoD is expected to spend approximately $45 billion a year on software development. However, according to Dr. Patricia Sanders, Director of Test, Systems Engineering and Evaluation, Office of the Under Secretary of Defense, Acquisition and Technology, in her 1998 Software Technology Conference keynote address, 40 percent of DoD's software development costs are spent on reworking the software. By the year 2000, this will equate to an annual loss of $18 billion. Additionally, a 1995 Standish Group survey,[3] resulting in 365 respondents and representing 8,380 software applications, produced the following results:

- 52.7 percent of projects were challenged, meaning they were completed

FIGURE 1. **Weapon System Software Sizes**

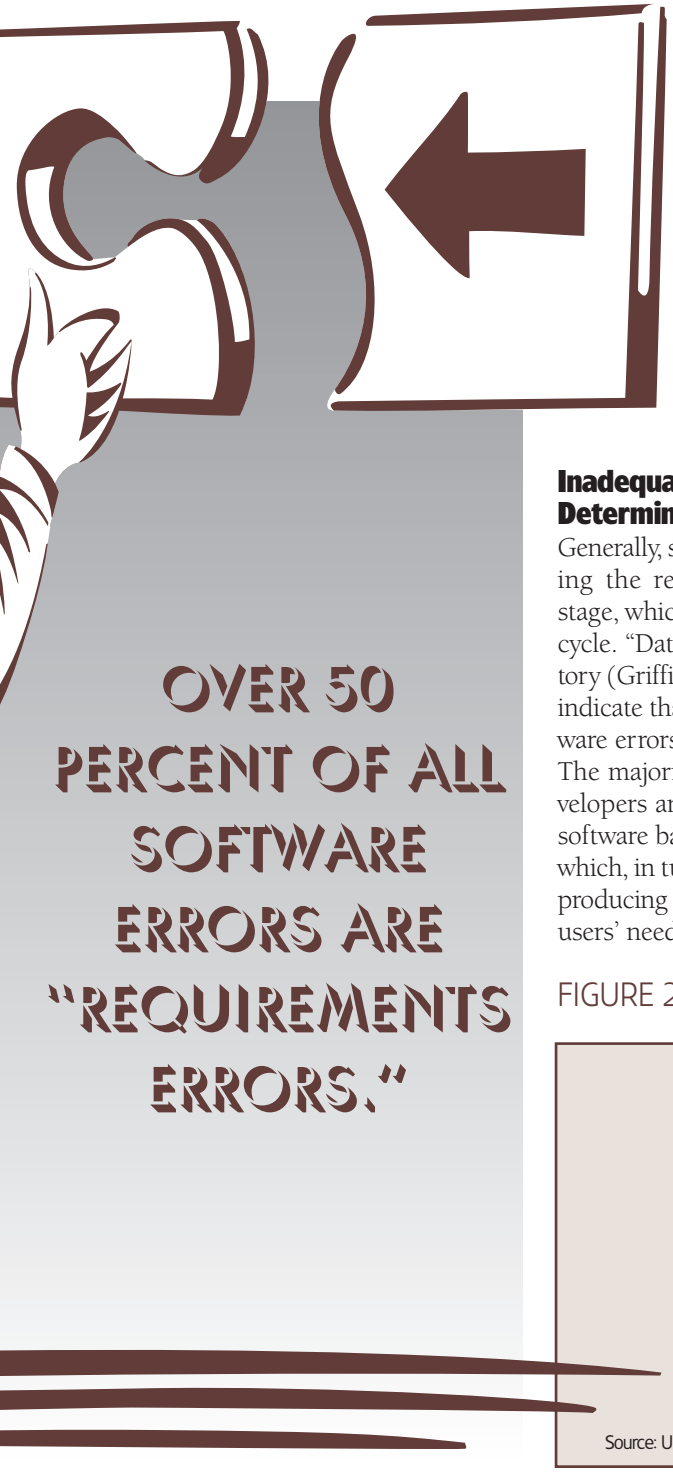
Weapon System	Source Lines of Code
M1 Tank	600,000
Scout/Cav	1,000,000
M2 Infantry Fighting Vehicle	1,560,000
Crusader	1,800,000
F-22	1,960,000
Aegis	2,840,000

Source: Dr. Delores Etter, Deputy Under Secretary of Defense, Science and Technology, Keynote Address, Software Technology Conference, 1999.

Nelson *is a software acquisition management and software cost estimating instructor, Army Logistics Management College (ALMC), Fort Lee, Va. He holds a master's degree in Management Information Systems from the University of Memphis.* ***Clark*** *is an acquisition instructor, ALMC. He holds a master's degree in Administration from Central Michigan University.* ***Spurlock*** *is a software cost estimating and statistics instructor, ALMC. She holds a master's degree in Mathematics Education from Virginia State University and a master's degree in Systems Engineering from the University of Virginia.*

but incurred cost and schedule overruns, resulting in fewer features than originally specified.
- 31.1 percent of projects were canceled at some time during the development cycle.
- The average cost overrun for challenged and canceled projects was 189 percent of the original cost estimate.
- The average schedule overrun for challenged and canceled projects was 222 percent of the original time estimate.
- On average, challenged projects were delivered with only 61 percent of the originally specified features and functions.

Furthermore, Sanders stated that only 16 percent of software development projects would finish on time and on budget.



OVER 50 PERCENT OF ALL SOFTWARE ERRORS ARE "REQUIREMENTS ERRORS."

This article focuses attention on two major causes of software and cost estimating problems: inadequate requirements determination and inadequate software cost estimates. Besides focusing attention on these two important areas, we present readers proposed solutions to inherent problems associated with software requirements and cost estimating.

Inadequate Requirements Determination

Generally, software problems begin during the requirements determination stage, which starts in Phase 0 of the life cycle. "Data collected at Rome Laboratory (Griffiss Air Force Base, New York) indicate that over 50 percent of all software errors are 'requirements errors.'"[4] The majority of the time, software developers are designing and developing software based on faulty requirements, which, in turn, results in the developers producing software that does not satisfy users' needs.

A 1994 survey conducted by IBM's Consulting Group found that 88 percent of the large software-intensive systems being developed by 24 leading companies, "had to be substantially redesigned."[5] This repeating problem results in 40 percent of DoD's annual software development dollars, or a whopping $18 billion spent on reworking the software. Furthermore, DoD knows from experience that correcting software errors follows the "1-5-100" rule.[6]

For example, a software error costing a mere $1 when caught early in the life cycle, costs $5 to correct at midpoint and $100 to correct later in the life cycle. What can DoD do to solve this problem?

We believe the user representative must prepare a Users' Functional Description (UFD) prior to Milestone I for all software-intensive systems, which includes all automated information systems, all C4I systems, and most, if not all, of today's major weapon systems. According to U.S. Army Training and Doctrine Command Pamphlet 71-9, *Force Development Requirements Determination*, August 1998, the UFD is intended as a refinement of the operational requirements for information technology (IT) capabilities contained in the Operational Requirements Document. As such, the UFD is the user representative's tool for effectively and completely communicating IT requirements to the program manager (PM).

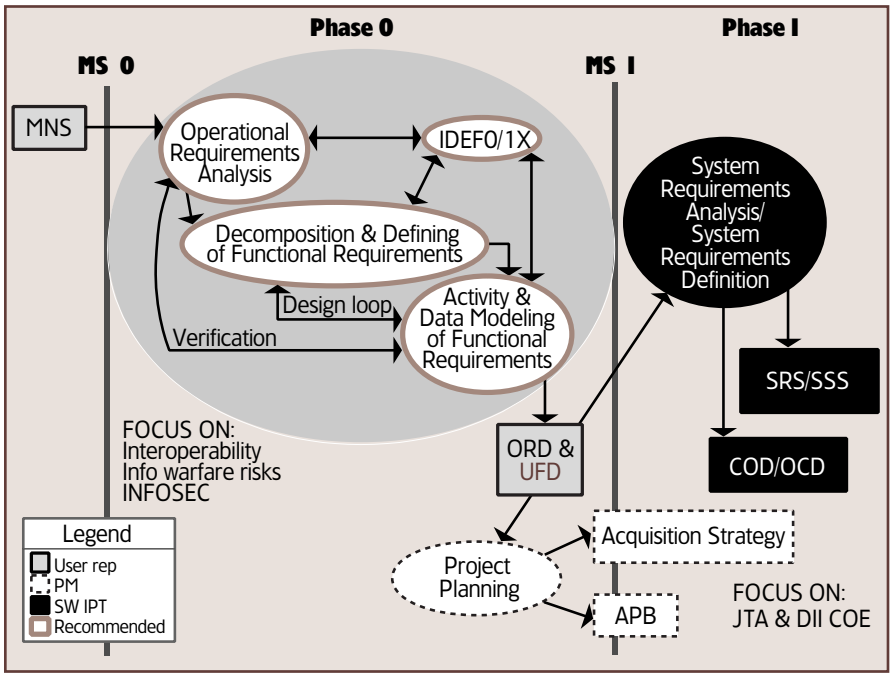
The power of the UFD lies in its use of Integrated Computer Aided Manufac-

FIGURE 2. **Weapon System Software Dependencies**

Weapon System	Year	Percent of Functions Performed in Software
F-4	1960	8
A-7	1964	10
F-111	1970	20
F-15	1975	35
F-16	1982	45
B-2	1990	65
F-22	2000	80

Source: U.S. Air Force, "Bold Stroke" Executive Software Course, 1992.

FIGURE 3. **Proposed Phase 0 Activities for Software Acquisition Management**



Source: Jim Clark and Army Maj. Mike Nelson.

turing Definition (IDEF) models to communicate IT operational requirements to the PM. Using the Army format as an example, the user representative describes the functions and activities that IT must perform for the user community via the Joint Technical Architecture (JTA)-mandated IDEF0 activity modeling process, which is the version of IDEF that applies and defines the standard activity modeling. The user representative also describes data requirements necessary to support the functions and activities using the JTA-mandated IDEF1X data modeling process, which is the version of IDEF that applies and defines the standard for data modeling. (For more information on IDEF0 and IDEF1X modeling, refer to Federal Information Processing Standards, Publications 183 and 184 respectively.)[7]

The problem within the Army, and perhaps most of DoD, is the requirement to prepare the UFD is left to the discretion of the user representative. DoD guidance concerning the use of modeling to define user requirements is contained in the JTA and the Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework. According to the JTA, if activity or data models are going to be developed for a system, then they must be developed in accordance with the IDEF0 and IDEF1X standards respectively. The C4ISR Architecture Framework identifies activity and data models as nonessential supporting framework products. Therefore, no mandate exists for the user's representative to apply these modeling techniques to better define the operational requirements and communicate them to the PM! Herein lies the crux of the problem.

A Step in the Right Direction

DoD has taken a step in the right direction by adding a required modeling and simulation element to the acquisition strategy in the proposed Change 4 to DoD 5000.2-R.[8] However, this action alone will not solve our inadequate requirements determination problem.

The acquisition strategy is prepared by the PM, but the PM is not responsible for defining the operational requirements. The PM is responsible, with the help of the user representative, for converting the operational requirements into system requirements that satisfy as many of the operational requirements as possible without violating the "building codes" mandated by the JTA and the Defense Information Infrastructure Common Operating Environment. The PM's use of modeling and simulation during the system requirements analysis process [software terminology] should prove very useful in converting operational requirements into system requirements. But, if the user representative fails to properly communicate operational requirements to the PM, system requirements will be converted incorrectly. The old computer adage of "garbage in equals garbage out" becomes a reality.

Hence, the requirement to use modeling and simulation during requirements determination needs to be moved back to the operational requirements analysis process in the form of a UFD. As depicted in Figure 3, the IDEF0 and IDEF1X models contained in the UFD provide the bridge between Phase 0-developed operational requirements contained in the Operational Requirements Document, and Phase I-developed system requirements documented in the System Requirements Specification and the Concept of Operations Description. We believe the absence of this bridge is the cause of the inadequate requirements determinations.

Recommendation

We recommend that a requirement, mandating the user representative prepare a UFD prior to Milestone I, be added to DoDD 5000.1,[9] along with a standard format for its preparation.

Our proposed strategy will also provide the PM with information needed to prepare the initial software cost estimate required at Milestone I as part of the overall acquisition strategy and the acquisition program baseline. Coupled with our proposed mandate for user representatives to prepare UFDs is the implied mandate for DoD to provide them with the personnel and training necessary to make the mandate a reality. A less preferred alternative to training and staffing user representatives to perform this critical function is DoD funding for outsourcing or privatization of UFD development.

Of course, the recommendations discussed here call for an aggressive investment of money and effort. However, the potential return on these investments, which includes annual cost savings of $18 billion, makes these recommendations well worth consideration.

Inadequate Software Cost Estimates

According to Dr. Delores Etter, Deputy Under Secretary of Defense for Science and Technology, half of all DoD software projects end up costing twice as much as originally estimated.[10] Why? We find two major reasons.

The first reason is incorrect software size estimates. The key to developing accurate software cost estimates is correctly estimating the size of the software. Since the PM determines size of the software based on the users' requirements, why are our software size estimates incorrect? Three reasons invariably surface:

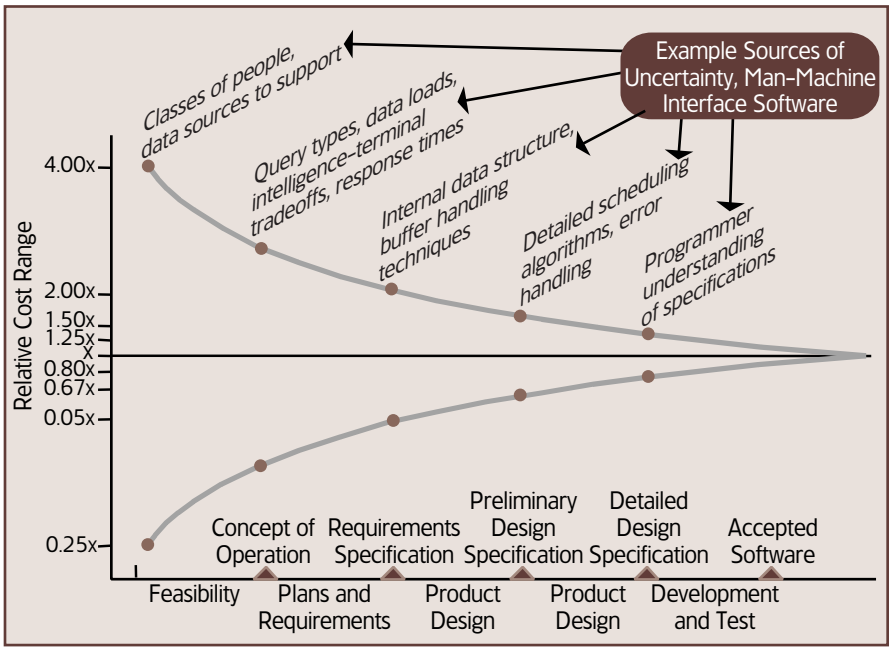
- User representatives are incorrectly and incompletely defining requirements.
- Program managers are using the wrong software size measure.
- A lack of training pervades DoD in the areas of software sizing and cost estimating.

Now that we have discussed and offered a solution for the problem of inadequate requirements determination, we will focus next on the problem of using the wrong software size measure, followed by a discussion on the lack of software sizing and cost estimating training.

Software Size Measure

In the past, most parametric models determined software costs based on an estimated number of source lines of code. Figure 4 vividly depicts the extremely high error rate (up to 400 percent) associated with estimating software costs, using source lines of code-based parametric software models, early in the life cycle. The problem lies not with the accuracy of the algorithms in the models, but with the inaccuracy of the size measurements fed into the models.

FIGURE 4. **Typical Accuracy of Source Lines of Code-Based Software Cost Estimation by Development Phase**



Source: Barry W. Boehm: Software Engineering Economics.[11]

> THE [ERROR RATE] PROBLEM LIES WITH THE INACCURACY OF THE SIZE MEASUREMENTS FED INTO THE MODELS.

Why have source lines of code proven to be an ineffective software size measure early in the life cycle (Phase 0/Milestone I)? Careful examination reveals several reasons:

- Programming languages have varying numbers of instructional units used to process a command. For example, to execute a command in Ada might require several hundred source lines of code, while the same command in C++ could require only several source lines of code. Further complicating this issue is the fact that prior to Milestone I, when the initial software cost estimate is being developed, the programming language(s) is not yet determined.
- Source lines of code estimates are based on a technical or physical view of the system, which does not exist at Milestone I. While historical source lines of code counts may be available from prior projects, their reliability is suspect, especially if the development language(s) is different or unknown.
- No standard definition or method exists for measuring source lines of code.

A more appropriate software size measure, at least in the early phases of the life cycle, is function points (FP). Function Point Analysis (FPA) is a method for determining software size based on function points, which A. J. Albrecht of IBM first introduced in 1979. Today, the nonprofit International Function Point Users Group (IFPUG) upholds the FPA methodology.

The FPA method boasts a high success rate and is compatible with the Institute of Electrical & Electronics Engineers/

Electronics Industries Association 12207, an industry implementation of an international standard for software life cycle processes; International Organization for Standardization/International Electrotechnical Commission 15504, an international standard for software process improvement practices; and several commercially available software cost estimating models. FPA is concerned with determining the number of logical, i.e., user identifiable, external inputs, external inquiries, external outputs, external interface files, and internal logical files that the software will contain. Each instance of these function types is assigned a standard number of FPs based on its class and complexity. The FPs are then totaled to determine the size of the software.[12]

The beauty and power of FPA is that each of these five function types maps directly to the information contained in the IDEF0/1X models that the user representative would provide to the PM.

In other words, the IDEF0/1X models provide all the information required for determining the initial FP count of the software to be developed. This FP count then becomes the size measurement that can be fed into an FP-based parametric model, which, in turn, produces the software cost estimate in terms of work effort and cost. Because FPA is based on a functional or logical view, instead of a technical or physical view of the system, it also allows the PM to accurately quantify software size in terms the user representative and the program manager can understand.

Currently, the majority of DoD software cost estimates are not derived using FPA, and few, if any, DoD cost estimators are IFPUG-Certified Function Point Specialists. The Army's Cost and Economic Analysis Center, and DoD's Cost Analysis Improvement Group, reported only a few software-intensive systems used FPA as the basis for deriving software cost estimates. Unfortunately, the majority of software cost estimates continues to be based on source lines of code, and will experience the problems discussed earlier.

RECOMMENDATION

We recommend that DoD discontinue using source lines of code and begin using FPs as the Department's primary software size measure.

The number of source lines of codes contained in the weapon systems listed in Figure 1 indicates the large majority of Acquisition Category I/IA programs contains between 10,000 and 100,000 FPs. According to Capers Jones, Chairman of Software Productivity Research (SPR), the average cost per FP for military systems with 10,000 and 100,000 FPs is $11,232 and $16,161, respectively.[13] Considering that, we simply can not afford to continue source lines of code "guesstimating" to approximate size of the software in these systems. Nor can we rely on DoD contractors to provide the sole software size estimate. User representatives must accept the responsibility to accurately and completely define IT requirements of systems; likewise, PMs too must assume responsibility for converting IT requirements into accurate and complete software cost estimates.

According to Mike Cunnane, SPR, an experienced IFPUG-Certified Function Point Specialist could count 1,000 FPs per day. Based on the government rate of $1,573 per day, hiring one of SPR's IFPUG-Certified Function Point Specialists to count the FPs for a 10,000-FP system would cost approximately $15,730. Keeping in mind that source lines of code-based software cost estimates for a system this size can be off by as much as $84 million (as depicted in Figure 4), $15,730 would seem a bargain price for obtaining an accurate software size measurement.

An added benefit of using FPA is its well-defined status and recognition as an internationally governed unit of measure. Essentially it provides a common language for the PM to communicate the size of the effort to prospective contractors. Invariably, programming languages will differ in the number of source lines of code required to perform the same function or to execute the same command. Contractors also will differ in the programming languages they use. One contractor might favor C++, another Ada, and a third, Visual Basic. The waters are further muddied by the fact that no universally accepted standard definition exists for a source line of code.

FPA also provides an established set of rules for applying its internationally governed unit of measure, which is completely independent of the programming language or database management system used to develop the software. Based on the 1990 Massachusetts Institute of Technology study conducted by C.F. Kemerer, given the same set of requirements, the FP counts provided by the government and contractor IFPUG-Certified Function Point Specialists, should be within 10 percent of each other.[14] The results of the study were based on the 1990 IFPUG CPM Release 3.0, which had many fewer clarification and counting rule examples than the current IFPUG CPM Release 4.1.

Therefore, as Carol Dekkers, President of IFPUG, recently stated, "It is absolutely logical to conclude that if the study were done today with the IFPUG 4.1 rules, that the conclusions would be the same, but with much greater accuracy." FPA puts everyone on the same level playing field. With common agreement on the size of the software, the PM's task essentially becomes evaluating which contractor can produce the required FPs with the least number of defects for the least amount of money.

Why are we doing so poorly in estimating our software costs? Today, the availability of training is certainly no excuse because the Defense Acquisition University (DAU) provides such training. The basis for this training requirement is not in guidance, but in law! Congress passed the Defense Acquisition Workforce Improvement Act (DAWIA) of 1990 because it tired of hearing bad publicity about DoD on "60 Minutes," in *The Washington Post*, and from other media.

Immediate Need For Cost Estimating Training

Since 1995, DAU has sponsored a two-week course in Software Cost Estimating (BCF 208). To our knowledge, this

is the only DoD-sponsored course covering software cost estimating in detail. Other DAU courses (ACQ 201, Intermediate Systems Acquisition Course; IRM 101, Basic Information Systems Acquisition Course; BCF 101, Fundamentals of Cost Analysis) touch on software cost estimating, but BCF 208 covers the full range of software acquisition management and software cost estimating, addressing FP- and source lines of code-based parametric models.

DoD students may attend DAU courses tuition-free, while contractor personnel employed under a current contract with the government can attend BCF 208, Software Cost Estimating, for the current rate of $66.00 per day.

Expense, or lack of available training, are not realistic excuses for not "growing" our own quality software cost estimators. Perhaps because the training is considered an "elective," too few people view it as career-enhancing.

RECOMMENDATION

We recommend software cost estimating training, such as BCF 208, be mandatory for DAWIA certification for all individuals involved in acquisition career fields such as Program Management; Business, Cost Estimating, and Financial Management; and Computers and Communications; as well as for individuals working combat development issues.

To date, 310 students, representing the budgeting, contracting, and auditing communities, have attended BCF 208. Certainly, practitioners in these career fields require training in the software cost estimating process, but personnel working in other functional areas and career fields also need to be part of the software cost estimating process. Understandably, estimating software costs can be a difficult, daunting challenge for those not well-versed in software issues.

The Software Technology Conference (STC), DoD's premier annual conference on software, did not offer a cost estimating track in past conferences. STC should not miss another opportunity to provide this critical training during the next conference, scheduled for May 2000 in Salt Lake City, Utah.

The second major reason why half of all DoD software projects end up costing double the original estimate, and why DoD software projects suffer an average schedule slippage of three years,[15] is requirements instability. In other words, once the software size estimate is determined, its stability is tenuous at best. Three major factors emerge as the culprits: requirements creep, technology insertion, and regulatory changes.

THE MAJORITY OF SOFTWARE COST ESTIMATES CONTINUES TO BE BASED ON SOURCE LINES OF CODE, AND WILL EXPERIENCE PROBLEMS.

Requirements Creep

Internal and external forces generate requirements creep. If the user representative fails to determine completely and accurately the requirements up-front, then this internal force will add new and change existing requirements throughout the development of the software. A system required to interface with another system experiences externally generated requirements creep every time the other system changes either its technical or functional external interface requirements.

Technology Insertion

Technology insertion has a software component as well as a hardware component. For example, almost all of DoD systems are today running on a commercial operating system. As vendors release major upgrades to their operating systems, they stop supporting their previous versions. The PM must then either purchase a very expensive maintenance contract, somehow start internally maintaining the current version of the operating system, or upgrade to the vendor's new version of the operating system. Upgrading often means the PM must reconfigure the software so it operates properly and efficiently with the new version of the operating system. At a minimum, the PM must conduct extensive testing to ensure the software still functions correctly. Inevitably, the problem multiplies as the PM must integrate more and more commercial-off-the-shelf products into the software inventory.

The hardware side of this coin is not much different. Peripheral devices, such as printers, scanners, and barcode readers require software drivers to operate properly. These devices may also contain non-modifiable firmware, wholly incompatible with the current system, which, in turn, forces the software developer to change the application software so it too functions correctly with the new device. Major hardware platform upgrades or changes can require extensive testing to ensure the software functions normally, with no residual adverse effects.

Regulatory Changes

Regulatory changes are probably the most overlooked reason for requirements instability, but they have a big impact, especially on management information systems. DoD management information systems usually implement the business processes for a specific functional area, such as finance, supply, maintenance, and property accountability. Regulations define and govern these business practices, processes, and procedures. What happens when the "powers that be" decide to change the regulation within a certain functional area? Obviously, the software within the management information system that implements that regulation must also be changed.

Is there a solution to the problem of requirements instability? Some would argue it remains doubtful that this problem can be, or even should be, solved. However, DoD can certainly take steps to alleviate the adverse cost and schedule impacts caused by requirements instability.

RECOMMENDATION
We recommend DoD apply the industry-proven and accepted rule-of-thumb for requirements growth to all of its initial software cost estimates.

According to Jones, "The average growth of unplanned, unanticipated requirements is about 1 percent to 2 percent per month during the design and coding phases of typical software projects."[16] A 1-percent-per-month growth in requirements is also supported by the results of the recent study conducted by Jeff A. McDowell and Dr. Lewis S. Fichter of Tecolote Research, Inc., which they presented at the 1999 Software Technology Conference.

For example, consider a 10,000-FP software project, which we estimate will take 36 months to develop. (The 36 months does not include installation or fielding of the system.) Using the figure of $11,232 per FP, the initial software cost estimate for development of the project comes in at $112,320,000. Spread over a 36-month period, it equates to a staff productivity of 278 FPs per month. However, if the worst-case, 2-percent-requirements-growth rule-of-thumb is correct, this project actually ends up at 17,200 FPs, a cost of $193,190,400, and will take approximately 62 months to develop. By failing to consider the inevitable growth in requirements, our original estimate for this project almost doubled in cost, and the schedule slipped 26 months.

RECOMMENDATION
Using this same example, we recommend the following strategy for addressing unplanned, unanticipated requirements. Baseline the contract for the development of the project at 17,200 FPs, 62 months, and $193,190,400. However, since the requirements for the additional 7,200 FPs currently do not exist, hold them in management reserve. As new requirements come in, the PM sizes and prioritizes them, and then withdraws the appropriate number of FPs from the management reserve and passes them on to the contractor for development.

For the contractor to meet the original schedule, the PM must still establish a deadline for the incorporation of new requirements. This technique gives the PM much more latitude in dealing with the adverse effects of requirements instability on the cost and schedule of a project. By incorporating the requirements growth rule-of-thumb into the initial contract baseline, the PM can shift the deadline for incorporating new requirements much further to the right on the project's calendar, without affecting the project's original cost or schedule. Not only does this technique generate more accurate and realistic cost estimates, but it also gives PMs the flexibility required to better satisfy changing needs of their customers.

Now is the Time

The time for change is now as DoD continues losing $18 billion per year reworking its software, while only 16 percent of its software development is completed on time and within budget. To reverse this trend, DoD must do a better job defining operational requirements, estimating the size of the software, and cultivating well-trained software cost estimators. The training is available; the need is substantial; the timing is right.

Editor's Note: For questions, comments, or a copy of the references cited in this article, contact Nelson at **nelsonm@lee.army.mil.**

DoD Selects Vendors for Public Key Infrastructure Pilot

The Department of Defense has made its initial selection of vendors to enable secure, electronic business services with private industry.

Operational Research Consultants Inc., and Digital Signature Trust Co. are the first two candidates selected to supply the Department with Class Three Interim External Certification Authorities [IECA] for its public key infrastructure. This capability will allow DoD to electronically communicate with industry by enabling secure, private electronic business and paperless contracting. IECAs will be used to provide non-DoD personnel with certificate services compatible with the Department's public key infrastructure.

In May 1999 the Department released a solicitation for IECAs to support vendors conducting business with the Paperless Contracting Wide-Area Work Flow, Electronic Document Access, and Defense Travel System applications. Operational Research Consultants Inc., and Digital Signature Trust Co. are the first two candidates to successfully complete the testing, policy, and procedural requirements for IECAs. More IECA selections will be announced, as available.

Selection of these two vendors is a significant milestone in the rollout of the Department's public key infrastructure since it promotes broader industry participation. In addition, this pilot should provide additional data to refine the Department's requirements and procedures for use of future external certificate authorities.

Editor's Note: This information, published and released Sept. 21 by the Office of the Assistant Secretary of Defense (Public Affairs), is in the public domain at **http://www.defenselink.mil/news.**